Open in app ↗

Medium

Get unlimited access to the best of Medium for less than $1/week. Become a member

The Prettiest Session Hijack I Ever Fell For.

6 min read · Just now

Syed Muhammad Ahmad

Listen Share More

My mother is convinced that I am too naive for this world. Maybe we all are, or perhaps, being robbed by something beautiful is simply the oldest human tradition. In this article, ladies and gentleman, we will see how an Automattic owned app stole my fingerprints as I lent them a hand to shake.

Background

This week I faced an interesting dilemma. While exploring Mesh, a contact management app owned by Automattic, I was struck with a fortunate realisation. LinkedIn and Meta do not expose personal details like followers, following, and connections through their official APIs. Yet almost every contact management app fetches them. The question is, ***how?*** The answer is simple: **deception**. In this article I will scooby doo the mechanism by which these apps harvest your session tokens to make API calls that are, by any honest definition, identity theft.

The Facade

On paper, the app is simple. You make an account, you login with LinkedIn, Instagram, or other social media platforms, and all of your contacts get categorised and managed in a beautiful UI. I tried to do the same and very quickly I realised that there is no way to get information regarding our contacts and connections through official APIs. I tried to look for an optimistic answer, ***maybe a web scraper?*** No, social media apps ban scrapers heavily. ***Maybe some third party APIs?*** Still shady and unreliable. Every ***no*** forced me in a direction that I never wanted to go towards. Finally I decided it's time to go Batman, so I installed ***me.sh***, risked my socials for the betterment of society.

The Mechanism

I installed Mesh (formerly clay.earth) from App Store. I logged in with my email and the app asked to connect my LinkedIn and Instagram. The software engineer in me knew what was happening, but as a moth flies towards certain death, I flew towards identity theft. As I clicked integrate Instagram, a simple page appeared in front of me, ***"Login with Instagram",*** something we see daily, nothing new, nothing surprising, a simple social auth page. As soon as I clicked the button, my followers and following were inside Mesh. I knew what had just happened.

Mesh acted as ***Ahmad.***

The Evidence

The beautiful thing about chess is that you get your turn. So I decided to make my move by catching them in the act.

I set up Charles Proxy on my Mac, a tool that intercepts every network request your phone makes, decrypts it, and lays it bare in front of you. I configured my iPhone to route all its traffic through my Mac. I installed Charles's SSL certificate on my phone so it could break open encrypted HTTPS requests. Then I cleared everything, opened Mesh, and connected Instagram again. This time, I was watching.

The logs confirmed my suspicion. No calls to Instagram's official APIs, no calls to Mesh's own servers.

> *A simple call to Instagram's internal GraphQL endpoint.*

```
POST https://www.instagram.com/graphql/query
?query_hash=58712303d941c6855d4e888c5f0cd22f
```

> *That query_hash is essentially a secret menu item, a code Instagram's own engineers use internally that was never meant for outsiders.*

This is the same API that Instagram's own app uses to render the "following" screen. This endpoint is undocumented, unofficial, and completely off-limits for third party apps. How did Mesh find it out? By watching their own browser traffic in Chrome DevTools. To my non-engineering audience (if any), this can be done in 3 mouse clicks.

But finding the endpoint was the easy part. The real question was, ***what were they sending with it?!***

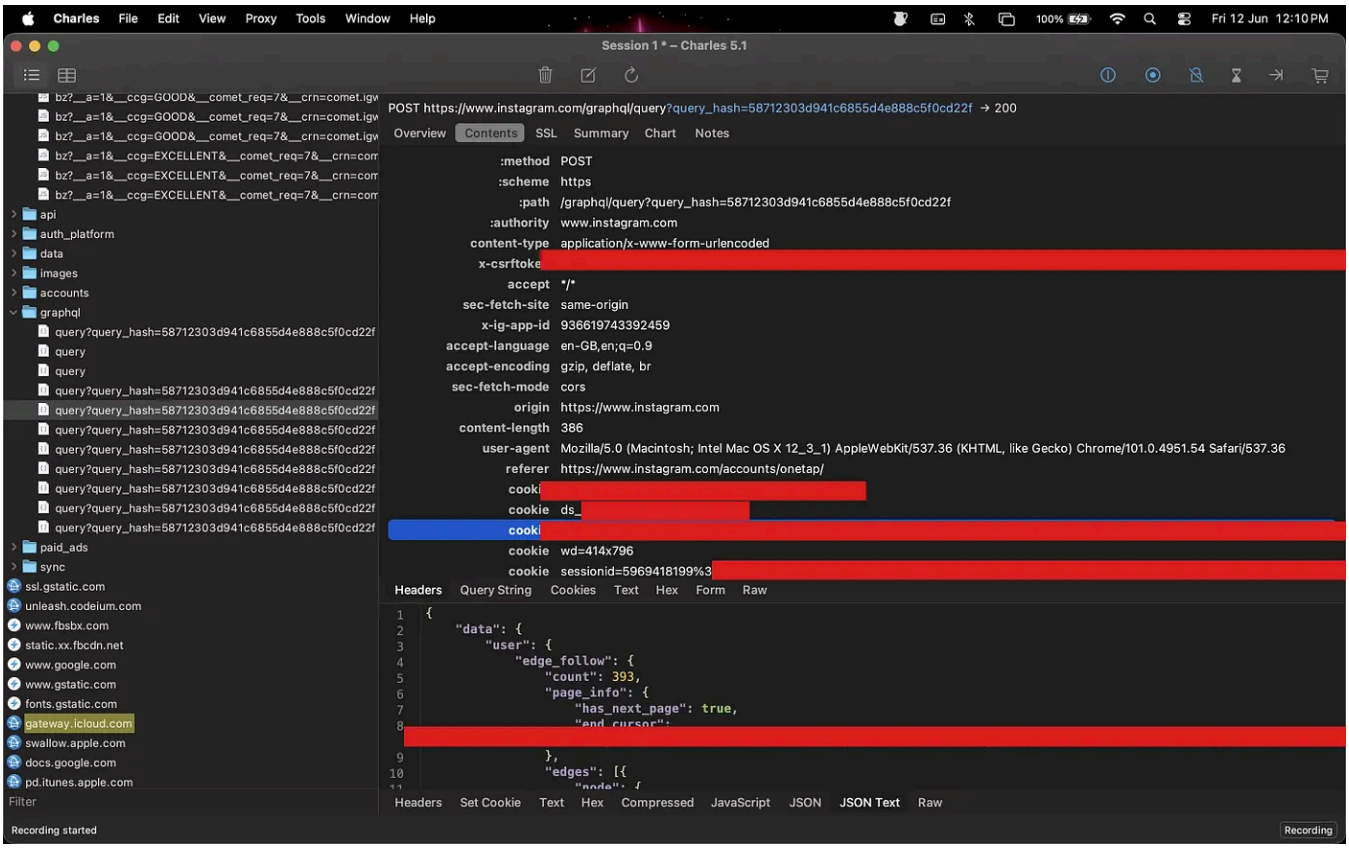
The Heist

Remember that simple login with Instagram page? Yeah, you guessed it right, it was never meant for social authentication. It was meant to harvest your session cookie and hijack your session.

The mechanism is painfully simple. Usually such social auth is done either via OAuth or by handing off the authentication to the OS, so that it can return a restricted token to you. iOS allows you to do this by using **ASWebAuthenticationSession.** Mesh bypassed this, they opened the authentication page using a plain **WKWebView.** It appears to you that Instagram's authentication page is opened in another sandbox, while Mesh controls this instance completely. Why did Mesh do this? Because they needed your password's evil twin, your actual session cookie, not the restrictive one which **ASWebAuthenticationSession** returns.

Using this session token, Mesh sends a request to Instagram's internal GraphQL API. Instagram thinks it's you who is in control. You think it's you who is in control, while it's Mesh playing you both simultaneously.

I won't make claims without any basis so see the attached screenshot of the network interception yourself.



The Instagram GraphQL call intercepted in real time. Sensitive information is hidden for common sense purposes.

See how the user-agent is a Mac Chrome user agent? Instagram's iOS app would never send that. Also if you can see just above the separator, you can read the words:

```
cookie sessionid=5969418199%3
```

… And the cookie doesn't expire when you close the app.

The Blast Radius

With all of this, there exists a bigger problem. They can do anything and everything. If someone has access to your session cookie, they have access to your account; ***they are you***. They can use this same cookie to send messages to anyone, post while acting as you, comment somewhere, follow people, unfollow people, all while you sit completely unaware.

The terrifying part is you will never know it happened. The people who receive those messages will never know either. And so it can very quickly become a case of reputation harm.

Reputations are built through ages. It takes just one tap to ruin everything.

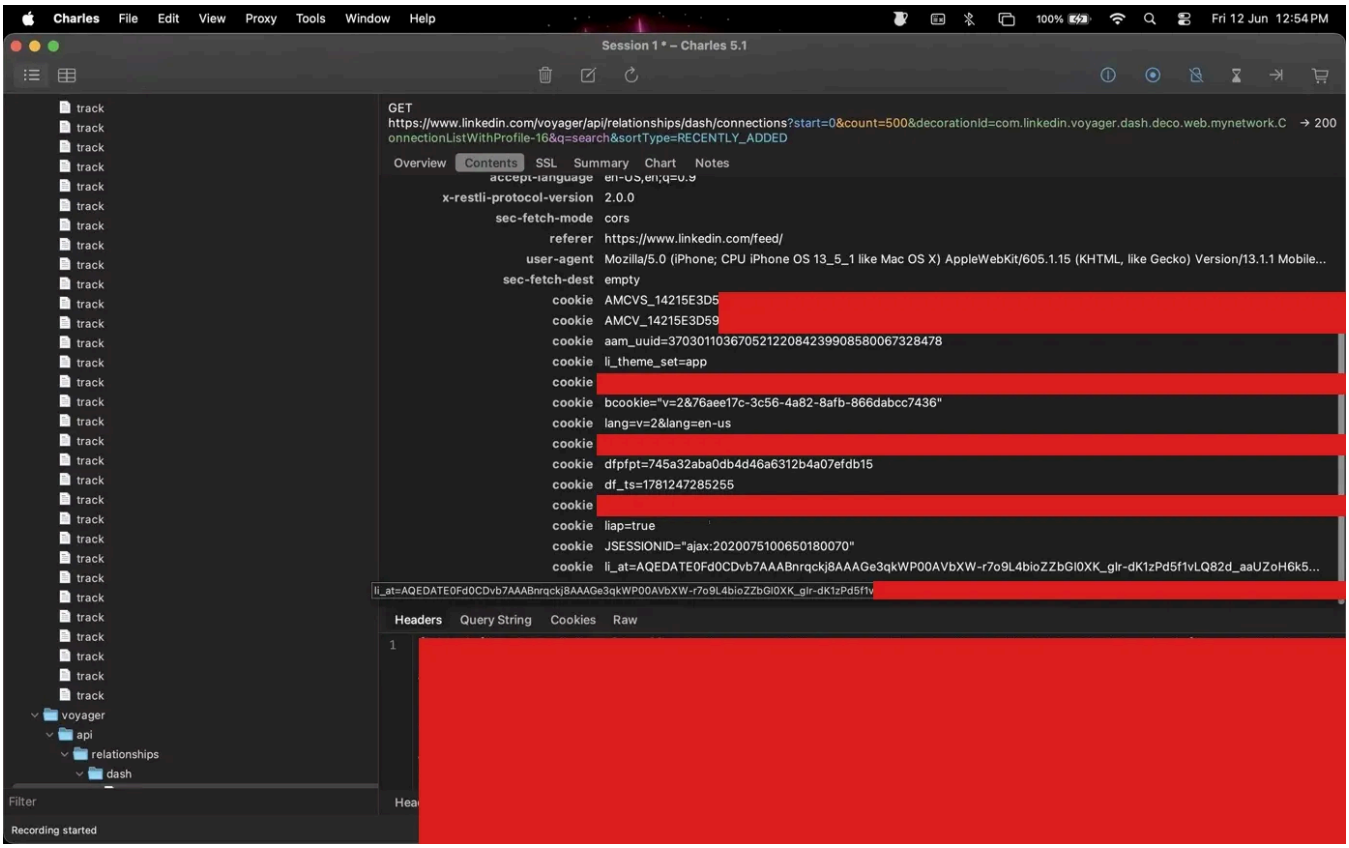
What about LinkedIn?

LinkedIn works the same way. You login while being inside Mesh's sandbox. They steal the session token. They send requests to LinkedIn's internal Voyager API and fetch your private data.

> *The endpoint:*
>
> `GET https://www.linkedin.com/voyager/api/relationships/dash/connections?count=500`
>
> *LinkedIn's own internal API, called with your stolen session cookie, returning 500 connections per request.*

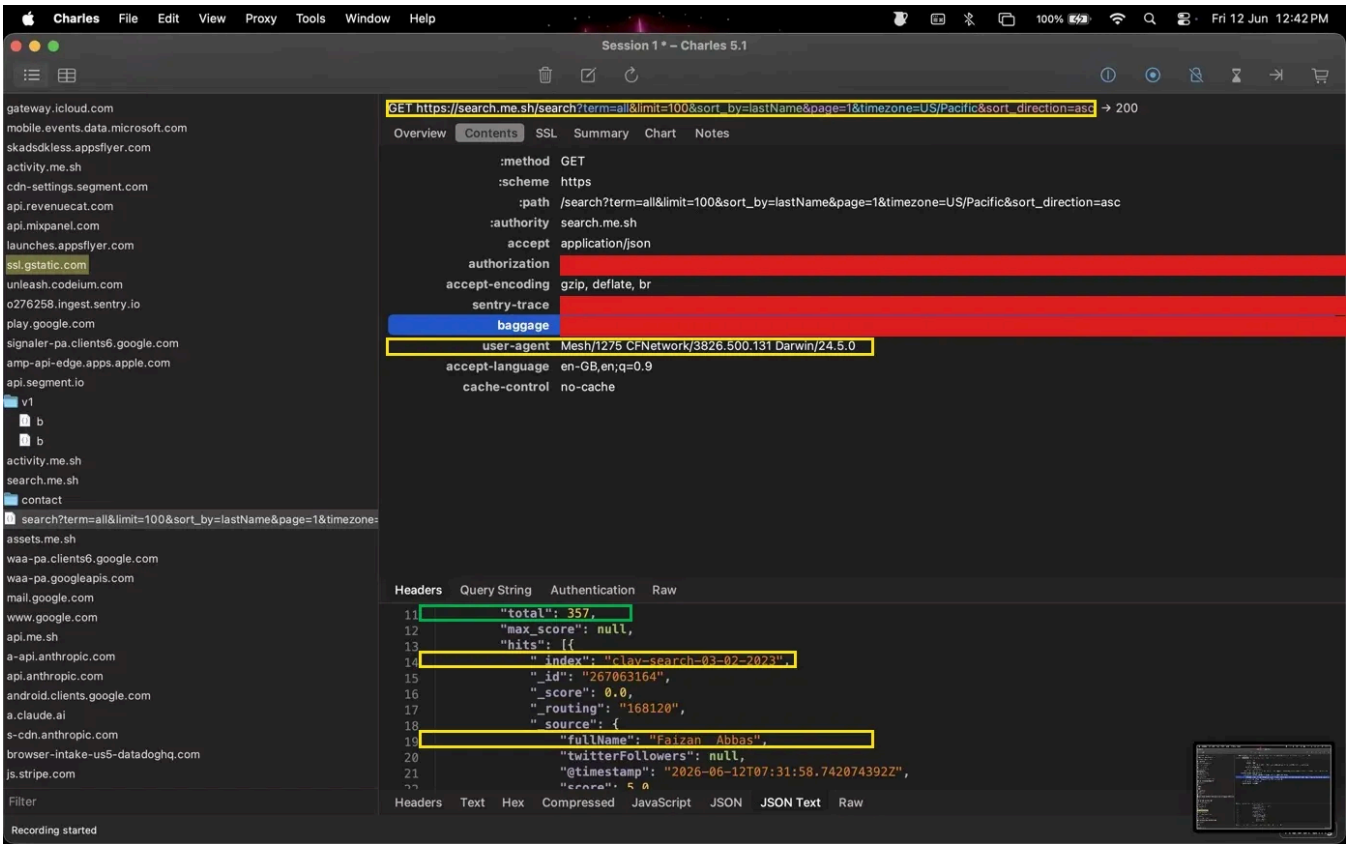
Imagine if they removed each and every one of your LinkedIn connections? Years gone down the drain. All because an app decided to be too smart.

For evidence, please take a look at the screenshot attached below. The **li_at token** is the evidence. It is basically LinkedIn's session token.



LinkedIn's internal voyager API called with my li_at token.

According to the official APIs, Mesh should never be able to get the data about my LinkedIn connections. But, here's a screenshot of Mesh's server returning the data of my connections after completing the LinkedIn integration. The only remaining argument they could make is that they used bots rather than session tokens. That argument dies against one fact: bots scrape public data. My private connection to Faizan Abbas is not public. The only way Mesh has it is through my authenticated session. The timestamps, the sequence, and their own server's response confirm it happened in a single afternoon. I rest my case



The user-agent is Mesh. The endpoint belongs to Mesh. The data returned is a real person from my network, someone who never signed up for Mesh, never consented to being in their database, but is sitting right there in their servers.

Note: This article gave you the knowledge, and now I want you to be responsible. Do not construct this attack or use it anywhere. If you do this, remember this is not moral no matter what you tell yourself. Actually, scratch that, who cares about morals anyways.

“Power tends to corrupt and absolute power corrupts absolutely.” — Lord Acton said this ages ago. The more I have read man, the more I am convinced that man is ultimately destined to follow the dark. The world is Machiavellian. Infrastructure that can be exploited, will be exploited. Perhaps by a rogue employee, perhaps by an ambitious executive or perhaps by a slight change in ToS policies. It is unfortunate that this hunt has etched paranoia in my being, and the man who is paranoid almost all the time is right almost all the time.

Morals? Yes, another pretty lie, another pretty illusion.

Session Hijacking | Cybersecurity | Mobile App Development | Cyber Security Awareness | Software Development

Edit profile

Written by Syed Muhammad Ahmad

74 followers · 9 following

A modern polymath.